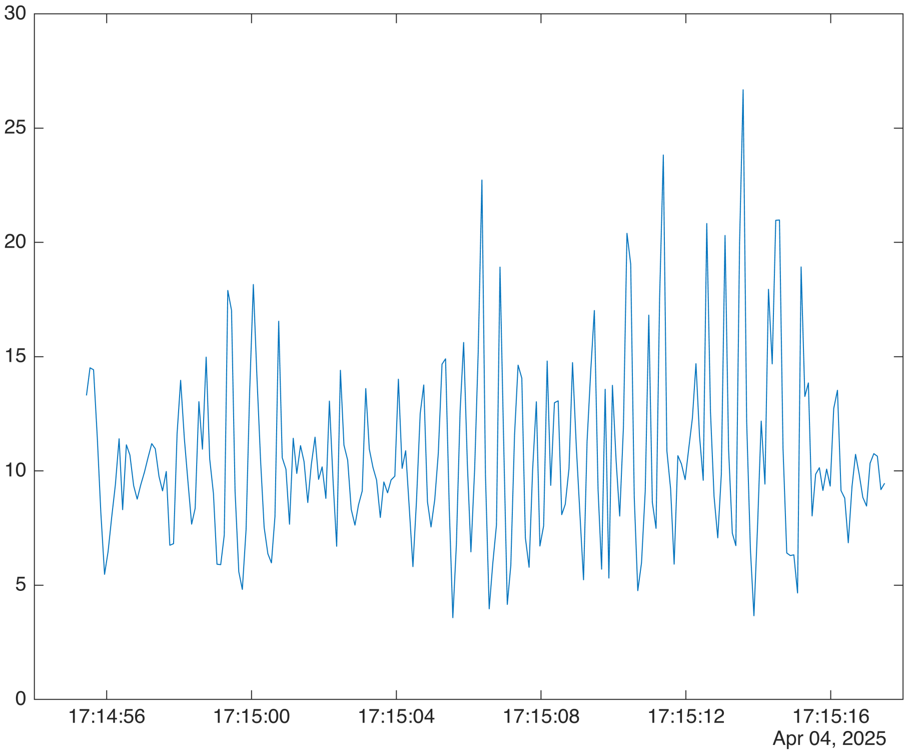
Implementing a Walking Step Counter using Accelerometer Sensor

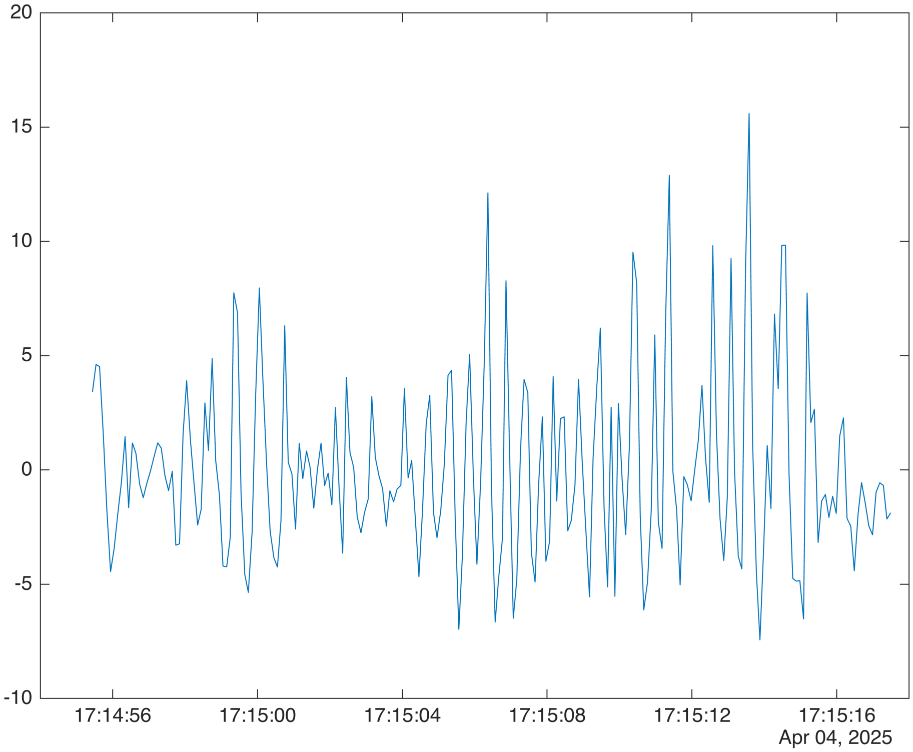
Step-1: Load Step signal and plot signal data

```
load hdjdkek.mat
% Extract X, Y, and Z columns
ax = Acceleration.X;
ay = Acceleration.Y;
az = Acceleration.Z;

% Compute Acceleration Magnitude
acc_magnitude = sqrt(ax.^2 + ay.^2 + az.^2);
fs=10;
% Add the computed magnitude to the timetable
Acceleration.Magnitude = acc_magnitude;

plot(Acceleration.Timestamp,Acceleration.Magnitude)
```



Step-2:Eliminate the trend in signal data

```
designal=detrend(acc_magnitude);
plot(Acceleration.Timestamp,designal);
```

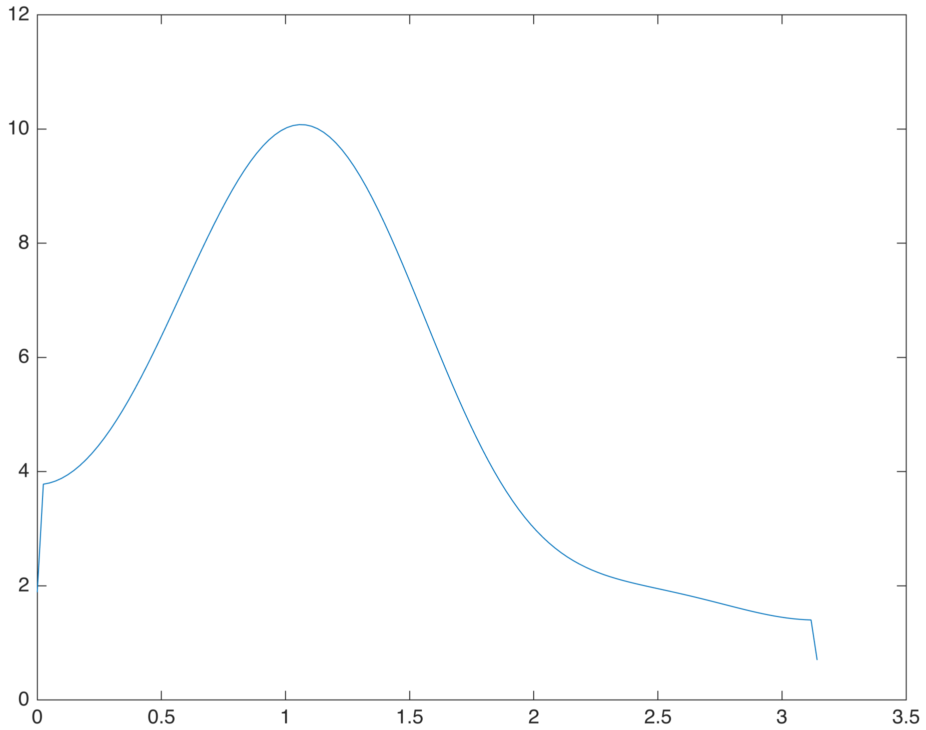


Step-3:Convert time domain to frequency domain

```
[pxx,fxx]=pwelch(designal,fs)
```

```
pxx = 129×1
    1.8871
    3.7810
    3.8016
    3.8359
    3.8838
    3.9452
    4.0200
    4.1081
    4.2092
    4.3232
      ⋮
fxx = 129×1
         0
    0.0245
    0.0491
    0.0736
    0.0982
    0.1227
    0.1473
    0.1718
    0.1963
    0.2209
      ⋮
```

```
plot(fxx,pxx)
```



Step 4-: Design low pass filter with automatic coding

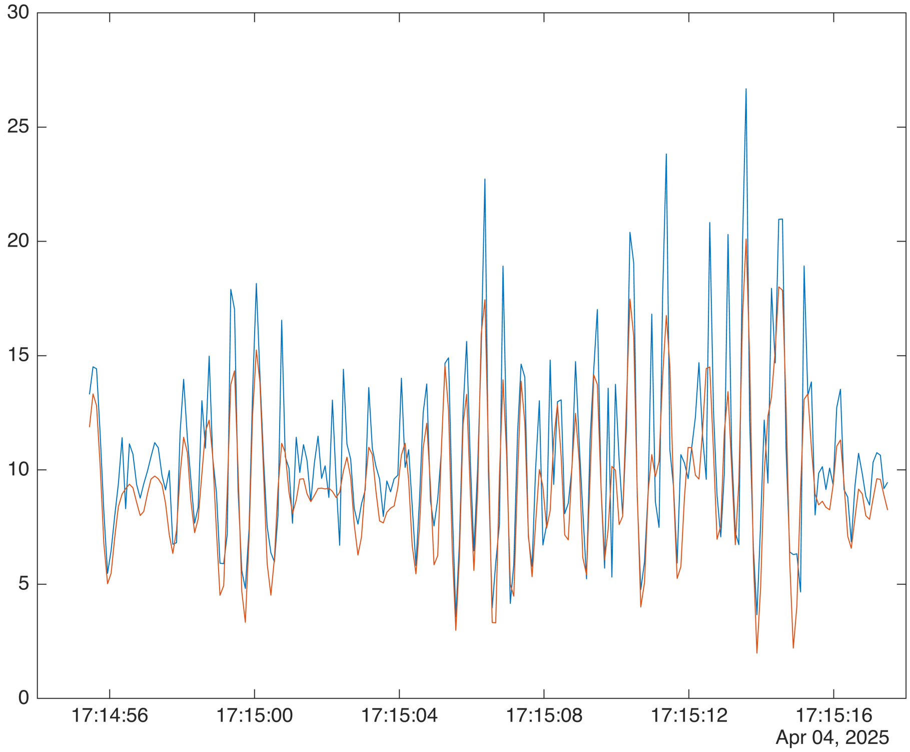
```
fs = 10;        % Sampling frequency (adjust if needed)
fc = 2;         % Cutoff frequency (Hz)
N = 4;          % Filter order
Rp = 0.5;       % Passband ripple (dB)

% Design Chebyshev Type I Low-Pass Filter
[b, a] = cheby1(N, Rp, fc/(fs/2), 'low');  % Normalize cutoff frequency
```

Apply Filter to the signal

```
filtered_data = filtfilt(b, a, acc_magnitude);  % Zero-phase filtering
figure;
```

Time Domain Visualisation

```
plot(Acceleration.Timestamp,Acceleration.Magnitude,Acceleration.Timestamp,fi
ltered_data)
```



Step 5:Peak finder

```
[peaks,locs]=findpeaks(filtered_data)
```

```
peaks = 38×1
   13.3283
    9.3762
    9.7344
   11.4321
   12.1804
   14.3390
   15.2497
   11.1689
    9.6160
    9.2059
      ⋮

locs = 38×1
     2
    12
    19
    27
    34
    41
    47
    54
    60
    65
     ⋮
```

Step 6:Total steps

```
xy=length(locs);
disp(xy)
```

```
    38
```